HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

ORIGINAL

PATENT APPLICATION

ATTORNEY DOCKET NO. ___200308341-1___

IFW

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors: Harish G. PATIL et al.

Confirmation No.: 7608

Application No.: 09/723,687

Examiner: A. J. Li

Filing Date: 11/28/2000

Group Art Unit: 2183

Title: BRANCH PREDICTION COMBINING STATIC AND DYNAMIC PREDICTION TECHNIQUES

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in **triplicate** is the Appeal Brief in this application with respect to the Notice of Appeal filed on ___05/13/2004___.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)
for the total number of months checked below:

( )	one month	$110.00
( )	two months	$420.00
( )	three months	$950.00
( )	four months	$1480.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of ___$330.00___. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit:___06/08/2004___
OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number_____ on _____

Number of pages:

Typed Name: **Christina L. Paz**

Signature:_____

Rev 05/04 (Aplbrief)

Respectfully submitted,

Harish G. PATIL et al.

By _____

Jonathan M. Harris

Attorney/Agent for Applicant(s)
Reg. No. **44,144**

Date: **06/08/2004**

Telephone No.: **(713) 238-8000**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Harish G. PATIL et al.	§	Confirmation No.:	7608
		§		
Serial No.:	09/723,687	§	Group Art Unit:	2183
		§		
Filed:	11/28/2000	§	Examiner:	A. J. Li
		§		
For:	Branch Prediction Combining Static And Dynamic Prediction Techniques	§ § § §	Docket No.:	200308341-1

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Date: June 8, 2004

Sir:

Appellants hereby submit this Appeal Brief in connection with the above-identified application. A Notice of Appeal was filed on May 13, 2004.

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, a Texas Limited Partnership, having its principal place of business in Houston, Texas, through its merger with Compaq Computer Corporation (CCC) which owned Compaq Information Technologies Group, L.P. (CITG). The assignment from the CCC to CITG was recorded on 01/16/2002, at Reel/Frame 012394/0118.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1-22.
Claim cancellations:	2, 3, 10, 11, 14, 19 and 20.
Added claims:	None.
Presently pending claims:	1, 4-9, 12, 13, 15-18, 21 and 22.
Presently appealed claims:	1, 4-9, 12, 13, 15-18, 21 and 22.

IV. STATUS OF THE AMENDMENTS

No claims were amended after the Final Office Action dated April 7, 2004.

V. SUMMARY OF THE INVENTION

Microprocessors fetch instructions from memory and execute the instructions. Some microprocessors have a pipelined architecture comprising various stages of processing (e.g., fetching, decoding, scheduling, executing, etc.). As such, multiple instructions can be processed through the pipeline in an efficient manner. For example, while one instruction is being scheduled, the next instruction can be fetched and decoded. Appellants' disclosure, pages 1-2.

A conditional branch instruction is an instruction that includes a condition that can be true or false. When the instruction is executed, the condition is checked to determine whether it is true or false. If the condition is true, program control jumps to a predefined set of instructions (the branch is said to be "taken"), whereas if the condition is false, control may continue with the instructions immediately following the conditional branch instruction (the branch is said to be "not taken"). Appellants' disclosure, page 2.

In this scenario, a problem occurs as to which instructions to fetch and place into the pipeline following a conditional branch instruction when the outcome of the instruction is not yet known. Appellants' disclosure, pages 2-3. To solve this problem, some microprocessors include a hardware-based branch predictor. Numerous variations of branch predictors are possible, but, in general, a branch predictor keeps a history of the previous executions of each conditional branch instruction and makes a prediction "on-the-fly" about a future execution of the branch instruction based on the historical information. For example, if the previous execution of a branch resulted in the condition being true, then the branch predictor may predict that the condition will be true the next time the branch instruction is fetched. The prediction may turn out to be correct or incorrect. "The branch predictor merely predicts the future outcome of the conditional branch instruction; the true outcome will not be accurately known until the branch instruction is actually executed. If the branch predictor turns out to have made the correct prediction, then instructions that must be executed are

already in the pipeline. If the prediction turns out to have been inaccurate, then the incorrect instructions that had been fetched must be thrown out and the correct instructions fetched. Performance suffers on mispredictions and increases on correct predictions. Choosing a branch prediction scheme that results in correct predictions much more often than mispredictions will result in the performance increase gained from correct predictions outweighing the performance hit on mispredictions." Appellants' disclosure, page 3.

"Aliasing" is a problem symptomatic of many dynamic, hardware-based branch predictors. Aliasing refers to the problem in which multiple branch instructions in a program may index to a common location in a table of predictions. Aliasing undesirably can cause the predictions for one branch instruction to be influenced by the outcomes of another branch instruction that indexes to the same location in a prediction table. Appellants' disclosure, page 4.

Appellants have addressed this issue by making use of static software branch prediction instructions such as instruction 208 in Figure 2. As disclosed by Appellants, such instructions, in at least some embodiments, comprise multiple groups of prediction bits. Each group is configurable to provide prediction information for a separate conditional branch instruction, such as any one or more of instructions 201-207.

VI.   ISSUE(S)

The issue in this Appeal is whether claims 1, 4-9, 12-13, 15-18 and 21-22 are patentable under 35 U.S.C. 103(a) over Mallick (U.S. Pat. No. 5,752,014) in view of Intel's "Intel® IA-64 Architecture Software Developer's Manual Volume 1:IA-64 Application Architecture" (hereinafter referred to as "Intel Volume 1") and in further view of Blaner (U.S. Pat. No. 5,649,178).

VII.   GROUPING OF CLAIMS

Appellants propose the following claim groupings:

(1)   claims 1 and 4-8 (claim 1 representative of this group);

(2)   claims 9, 12, 13, and 15-17 (claim 9 representative of this group);

(3)   claims 18, 21, and 22 (claim 18 representative of this group).

The claims in each group are patentable separately from the claims of the other groups and, as such, do not stand or fall with the claims of the other groups as argued below.

VIII.   ARGUMENT

A.   The Mallick Reference

As correctly noted by the Examiner, Mallick discloses a hardware dynamic branch predictor. See e.g., Figures 1 and 2. "The branch processing unit includes...a branch prediction unit for predicting the resolution of a conditional branch instruction utilizing the selected branch prediction methodology." See Abstract. The Examiner also correctly concedes that Mallick does not disclose the use of branch prediction instructions.

B.   The Intel Volume 1 Reference

Intel Volume 1 discloses the use of "separate Branch Prediction instructions (brp) where the entire instruction is hint information." Page 4-29. Each branch prediction instruction includes prediction information relative to only one conditional branch instruction. A displacement value is included in the branch prediction instruction to identify the particular conditional branch instruction referenced by the prediction information. Page 4-30.

C.   The Blaner Reference

Blaner provides an improvement to hardware-based conditional branch instructions predictors. The Background section of the Blaner patent explains that once a conditional branch instruction is initially executed, a history of execution of that instruction is established which is used to predict the result of subsequent executions of the same instructions. Various hardware branch prediction schemes exist. "One such scheme involves dynamic prediction of branch outcomes by tagging branch instructions in a cache with predictive information regarding their outcomes." The problem Blaner addresses is the loss of the predictive information when a cache line containing a conditional branch instruction is evicted from the cache, as is the nature of cache operation. If the evicted branch instruction is ever again brought back into the cache, establishing

the initial state of the instruction's predictive information is problematic. See col. 1, line 48 through col. 2, line 34.

As noted above, Blaner is directed to hardware, dynamic branch prediction logic. More specifically and with reference to Figure 1, the "[c]ache/BHT[1] 102 provides temporary storage for lines of data received from memory 100 and also provides branch history bits for branch instructions contained within the data. Thus, cache/BHT 102 provides for dynamic prediction of branch outcomes by tagging branch instructions in the cache with predictive information regarding their outcomes... ." Co. 3, lines 10-18.

Blaner's solution to his stated problem is the use of the "branch history cache" 103 depicted in Figure 1. Blaner's branch history cache 103 provides storage for evicted predictive information. As such, if a line containing a branch instruction is evicted from the cache, the associated predictive information tagged to that cache line is off-loaded to the branch history cache 103. Then, if that branch instruction is subsequently brought back into the cache, the instruction's predictive information is retrieved from the branch history cache 103 as well. See e.g., col. 2, lines 38-47 and col. 4, lines 4-21. Blaner thus provides an improvement to dynamic, hardware-based branch predictors.

D.   The Examiner erred in Rejecting Claim 1

Claim 1 requires, among other features, that "static branch prediction instructions comprise a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch instruction." The Examiner correctly concedes that Mallick does not teach separate static branch prediction instructions. Further, Intel Volume 1 does not teach a branch prediction instruction that comprises a group of prediction bits in which each group is configurable to provide prediction information for a separate branch instruction. At most, Intel Volume 1 discloses a branch prediction instruction that provides prediction information for only one branch conditional instruction. Intel Itanium Architecture Software Developer's Manual Volume 3: Instruction Set Reference (used by the Examiner in the first

---

[1] "BHT" stands for Branch History Table. Col. 3, lines 9-10.

Office Action and referred to as "Intel Volume 3") discloses a branch prediction instruction that includes an operand that "specifies the address of **the** branch instruction to which this prediction information applies." Intel Volume 3, page 3:28 (emphasis added). In referring to "the" branch instruction, the Intel Volume 3 document does not teach or suggest that each branch prediction instruction can provide prediction information regarding more than one branch instruction.

Blaner is also deficient in this regard. Blaner does not teach or suggest that "static branch prediction instructions comprise a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch instruction." Blaner does not even teach or suggest the use of processor-executed static branch prediction instructions. Instead, Blaner teaches the use of dynamic branch prediction hardware. At least for this reason, the Examiner erred in rejecting claim 1 and associated dependent claims.

E.   The Examiner erred in Rejecting Claim 9

Claim 9 refers to a static branch prediction instruction that provides "separate static branch prediction information about a plurality of conditional branch instructions." The Examiner correctly concedes that Mallick does not teach separate static branch prediction instructions. Further, Intel Volume 1 does not teach a branch prediction instruction that comprises prediction information about multiple conditional branch instruction. At most, Intel Volume 1 discloses a branch prediction instruction that provides prediction information for only one branch conditional instruction. Intel Itanium Architecture Software Developer's Manual Volume 3: Instruction Set Reference (used by the Examiner in the first Office Action and referred to as "Intel Volume 3") discloses a branch prediction instruction that includes an operand that "specifies the address of **the** branch instruction to which this prediction information applies." Intel Volume 3, page 3:28 (emphasis added). In referring to "the" branch instruction, the Intel Volume 3 document does not teach or suggest that each branch prediction instruction can provide prediction information regarding more than one branch instruction.

Blaner is also deficient in this regard. Blaner does not teach or suggest the quoted limitation above. Instead, Blaner teaches the use of dynamic branch prediction hardware. Thus, none of the art of record teaches or suggests a static branch prediction instruction that provides "separate static branch prediction information about a plurality of conditional branch instructions." Mallick and Blaner are not directed to branch prediction instructions and Intel Volume 1 discloses the use of a branch prediction instruction that can provide prediction information for only one branch instruction. For at least this reason, the Examiner erred in rejecting claim 9 and associated dependent claims.

F. The Examiner erred in Rejecting Claim 18

Claim 18 is directed to a method that requires, among other actions, "including a static branch predictor software instruction in a program, said branch prediction software instruction including branch prediction information configurable to pertain to a plurality of conditional branch instructions in the program." The Examiner correctly concedes that Mallick does not teach branch prediction instructions. Further, Intel Volume 1 does not teach a branch prediction instruction that comprises prediction information about multiple conditional branch instruction. At most, Intel Volume 1 discloses a branch prediction instruction that provides prediction information for only one branch conditional instruction. Intel Itanium Architecture Software Developer's Manual Volume 3: Instruction Set Reference (used by the Examiner in the first Office Action and referred to as "Intel Volume 3") discloses a branch prediction instruction that includes an operand that "specifies the address of **the** branch instruction to which this prediction information applies." Intel Volume 3, page 3:28 (emphasis added). In referring to "the" branch instruction, the Intel Volume 3 document does not teach or suggest that each branch prediction instruction can provide prediction information regarding more than one branch instruction.

Blaner is also deficient in this regard. Blaner does not teach or suggest the quoted limitation above. Instead, Blaner teaches the use of dynamic branch prediction hardware. Thus, none of the art of record teaches or suggests a branch prediction instruction that includes "branch prediction information
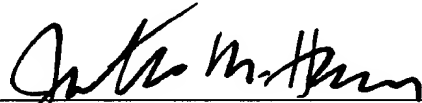
configurable to pertain to a plurality of conditional branch instructions." Mallick and Blaner are not directed to branch prediction instructions and Intel Volume 1 discloses the use of a branch prediction instruction that can provide prediction information for only one branch instruction. For at least this reason, the Examiner erred in rejecting claim 18 and associated dependent claims.

IX.   CONCLUSION

For the reasons stated above, Appellants respectfully submit that the Examiner erred in rejecting all pending claims. If any fees or time extensions are inadvertently omitted or if any fees have been overpaid, please appropriately charge or credit those fees to Hewlett-Packard Company Deposit Account Number 08-2025 and enter any time extension(s) necessary to prevent this case from being abandoned.

Respectfully submitted,

Jonathan M. Harris
PTO Reg. No. 44,144
CONLEY ROSE, P.C.
(713) 238-8000 (Phone)
(713) 238-8008 (Fax)
ATTORNEY FOR APPELLANTS

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
Legal Dept., M/S 35
P.O. Box 272400
Fort Collins, CO 80527-2400

APPENDIX TO APPEAL BRIEF
CURRENT CLAIMS

1.   (Previously presented) A computer system, comprising:

a processor that comprises a hardware branch predictor; and

software instructions executed by said processor, said software instructions comprising conditional branch instructions and separate static branch prediction instructions;

said static branch prediction instructions comprise a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch instruction.

2.   (Canceled).

3.   (Canceled).

4.   (Previously presented) The computer system of claim 1, wherein each group of static branch prediction bits comprises a pair of bits.

5.   (Previously presented) The computer system of claim 1 wherein said prediction information comprises a member selected from the group consisting of: do not use static prediction, predict taken, and predict not taken.

6.   (Original) The computer system of claim 4 wherein each pair of prediction bits corresponds to another instruction and each pair of prediction bits is encoded as: 00 and 01 mean do not use static prediction, 10 means predict taken and 11 means predict not taken.

7.   (Previously presented) The computer system of claim 1 wherein said static branch prediction bits comprise static branch prediction information that

comprises encoded information directing the processor to ignore the predictions supplied by the hardware branch predictor.

8. (Previously presented) The computer system of claim 1 wherein said hardware branch predictor comprises a log in which the results of all executed conditional branch instructions are stored.

9. (Previously presented) A processor, comprising:

fetch logic that fetches program instructions from a source external to said processor;

a dynamic branch predictor coupled to said fetch logic, said dynamic branch predictor supplies predictions regarding conditional branch instructions to said fetch logic;

an instruction queue coupled to said dynamic predictor, said fetch logic storing fetched instructions in said instruction queue; and

an execution unit coupled to said instruction queue and executing instructions provided from said instruction queue;

said fetch logic examines fetched instructions for a predetermined register identifier that identifies that instruction as a static branch prediction instruction that provides separate static branch prediction information about a plurality of conditional branch instructions.

10. (Canceled).

11. (Canceled).

12. (Previously presented) The computer system of claim 10, wherein said separate static branch prediction information for each conditional branch instruction comprises a pair of bits.

13.   (Previously presented) The processor of claim 9 wherein said prediction information comprises a member selected from the group consisting of: do not use static prediction, predict taken, and predict not taken.

14.   (Canceled).

15.   (Previously presented) The processor of claim 9 wherein said static branch prediction instruction comprises branch prediction bits that directs said fetch logic to ignore the predictions supplied by the dynamic branch predictor.

16.   (Previously presented) The processor of claim 9 wherein said dynamic branch predictor comprises a log in which the results of all executed conditional branch instructions are stored.

17.   (Original) The processor of claim 9 wherein said predetermined identifier comprises a register identifier.

18.   (Previously presented) A method of predicting the outcome of conditional branch instructions, comprising:

> including a static branch predictor software instruction in a program, said branch prediction software instruction including branch prediction information configurable to pertain to a plurality of conditional branch instructions in the program;
>
> fetching said branch prediction software instructions;
>
> decoding said branch prediction software instructions to determine if said decoded instruction is a branch prediction software instruction; and
>
> if said decoded instruction is a branch prediction software instruction, then using said branch prediction information for branch prediction.

19.   (Canceled).

20.   (Canceled).

21.   (Previously presented) The method of claim 18 wherein said branch prediction information comprises pairs of bits, each pair corresponding to another instruction.

22.   (Previously presented) The method of claim 21 further comprising decoding said pairs of bits to determine whether, for said other instruction corresponding to said pair, said other instruction is predicted taken, predicted not taken or no static branch prediction is provided.